MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

MODULE TEST 3 & 4

TRIMESTER 1, 2020/2021

ECE3166 ADVANCED MICROPROCESSORS

(All Majors)

23 OCTOBER 2020
3:00P.M. – 4:00P.M.
(1 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of SIX pages with TWO Questions only.
2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please print all your answers in your own answer papers.
4. Scan and submit your answer papers to Google Classroom.

Question 1

(a) Given that a code segment selector register is loaded with a value of 0123H. The base address of Global Descriptor Table (GDT) and Local Descriptor Table (LDT) are 1020 3000H and 4050 6000H, respectively.

[10203120H] = FFFFH	[40506120H] = 3FFFH
[10203122H] = 0900H	[40506122H] = 0D00H
[10203124H] = 9C08H	[40506124H] = 9B0CH
[10203126H] = 0701H	[40506126H] = 0A00H

i. Identify the Requested Privilege Level (RPL) and determine whether the descriptor is stored in the Global Descriptor Table (GDT) or Local Descriptor Table (LDT). [4 marks]

ii. Compute the address range of the code segment descriptor. [4 marks]

iii. Determine the base address and the size of the code segment. [5 marks]

iv. Is this code segment readable? [1 mark]

(b) If 32-bit paging is enabled on an IA-32 processor with a page size of 4KB. The linear address of an operand is 22100680H, while the contents of the corresponding page directory entry and page table entry are 10203007H and 04050005H, respectively.

i. Compute the physical address range of the page table and page frame. [4 marks]

ii. Determine the physical address of the operand. [3 marks]

iii. Justify whether write access is allowed if the operand is accessed from user mode. [4 marks]

Question 2

(a) Complete and reproduce Table Q2(a) in your answer paper.

Table Q2(a)

	Registers			
	EAX	MM0	XMM0	YMM0
Bits per register				

[4 marks]

(b) Out-Of-Order execution (OOOE) is a technique that allows a superscalar processor to execute multiple independent instructions concurrently. Write-After-Read (WAR) and Write-After-Write (WAW) are two false dependencies in OOOE. Show how register renaming can be used to solve WAR and WAW dependencies by completing and reproducing Table Q2(b) in your answer paper. Assume that the processor can execute up to **FOUR** instructions per cycle. [6 marks]

Continued …

Table Q2(b)

No.	Instruction	Cycle 1	Cycle 2
1	R1 ← mem1		
2	R2 ← R4 + R1		
3	R1 ← mem2		
4	R4 ← 7		
5	R3 ← R1 – R4		
6	R2← 4 + R5		

(c) Listing below shows manipulation of single precision floating point numbers using SSE instructions. The initial contents of register XMM0 and XMM1 are given in Table Q2(c).

```
struct SVector
{       float w, x, y, z;    // 4 bytes per variable
};
struct SVector SSE_Add(struct SVector op_a, struct SVector op_b)
{
        struct SVector ret_vector;
        int result = 0;
        __asm
        {
                MOVUPS XMM0,  op_a   // Move the vectors to SSE regs
                MOVUPS XMM1,  op_b
                MOVUPS XMM2,  XMM0
                MOVHLPS XMM2, XMM1
                ADDSS  XMM0,  XMM1   // Add scalar
                MOVUPS ret_vector, XMM0     // Save the return vector
        }
        return ret_vector;
}
int main()
{
        struct SVector obj1; struct SVector obj2; struct SVector res;
        obj1.w = 23.0; obj1.x = 12.0; obj1.y = 456.0; obj1.z = 67.0;
        obj2.w = 11.0; obj2.x = 87.0; obj2.y = 65.0; obj2.z = 32.0;
        // Call the function, SSE_Add.
        res = SSE_Add(obj1, obj2);
        printf("SSE Add: %f,%f,%f,%f\n", res.w, res.x, res.y, res.z);
        printf("\n");
        return 0;
}
```

Table Q2(c)

Instruction	Affected register = Content
MOVUPS XMM0, op_a	XMM0 = 42860000 43E40000 41400000 41B80000H
MOVUPS XMM1, op_b	XMM1 = 42000000 42820000 42AE0000 41300000H

Find the content of the destination register after execution of the `MOVHLPS` and `ADDSS` instructions. Show the contents in hexadecimal. [15 marks]

End of Paper

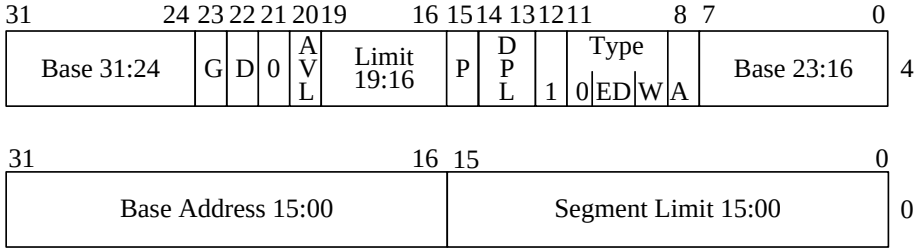
Appendix A

Summary of Selected SSE Instructions

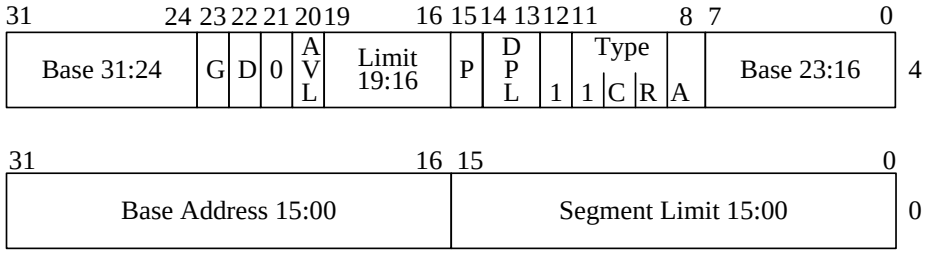
Mnemonic	Description
MOVAPS	Move aligned packed single-precision floating-point values
MOVUPS	Move unaligned packed single-precision floating-point values
MOVHPS	Move high packed single-precision floating-point values
MOVHLPS	Move packed single-precision floating-point high to low
MOVLHPS	Move packed single-precision floating-point low to high
MOVLPS	Move low packed single-precision floating-point values
MOVMSKPS	Move packed single-precision floating-point mask
MOVSS	Move scalar single-precision floating-point values
ADDSS	Scalar add pair of operands
ADDPS	Packed add pairs of operands
SUBSS	Scalar subtract pair of operands
SUBPS	Packed subtract pair of operands
MINPS	Packed return the minimum of the pair of operands
MAXPS	Packed return the maximum of the pair of operands

Appendix B

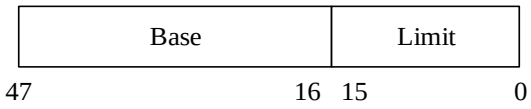
Data-Segment Descriptor:



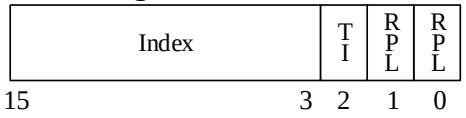
Code-Segment Descriptor:



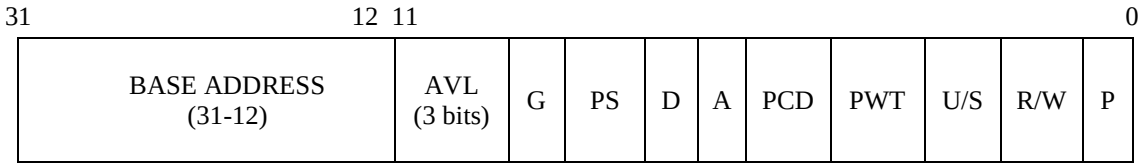
Global Descriptor Table Register (GDTR) and Interrupt Descriptor Table Register (IDTR):



Selector Register:



Page Directory or Page Table Entry:



Linear Address: